Lecture 1: Modelling and classification

Michael Patriksson

2011-10-26

Optimization, I

- "Optimum:" Latin for "the ultimate ideal;" similarly, "optimus:" "the best." To optimize is to bring something to its ultimate state
- Example problem 1: Consider a hospital ward which operates 24 hours a day. At different times of day, the staff requirement differs. Table 1 shows the demand for reserve wardens during six work shifts

Shift	1	2	3	4	5	6
Hours	0–4	4–8	8–12	12–16	16–20	20–24
Demand	8	10	12	10	8	6

Table: Staff requirements at a hospital ward

- Each member of staff works in 8 hour shifts. The goal is to fulfill the demand with the least total number of reserve wardens

A staff planning problem

$$
\begin{aligned}
\underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) := \sum_{j=1}^{6} x_j, \\
\text{subject to} \quad & x_6 + x_1 \geq 8, && \text{(work ends at shift 1)} \\
& x_1 + x_2 \geq 10, \\
& x_2 + x_3 \geq 12, \\
& x_3 + x_4 \geq 10, \\
& x_4 + x_5 \geq 8, \\
& x_5 + x_6 \geq 6, && \text{(work ends at shift 6)} \\
& x_j \geq 0, && j = 1, \ldots, 6, \\
& x_j \text{ integer}, && j = 1, \ldots, 6
\end{aligned}
$$

Optimization, II

- *Optimal solution*: $\mathbf{x}^*$, a vector of decision variable values which gives the objective function its minimal value among the feasible solutions
- Two optimal solutions: $\mathbf{x}^* = (4, 6, 6, 4, 4, 4)^{\mathrm{T}}$, $\mathbf{x}^* = (8, 2, 10, 0, 8, 0)^{\mathrm{T}}$
- *Optimal value*: $f(\mathbf{x}^*) = 28$

- The above model is a crude simplification of any real application
- Should add requirements on individual competence, more detailed restrictions, longer planning horizon, employment rules etcetera
- More complex models in practice

Example problem 2: the diet problem—description

- One if the first optimization problems studied in the 1930's and 40's. Motivated by the US Army's desire to feed field GI's at minimum cost
- Classic formulation by George Stigler (1939): for a moderately active person weighing 154 pounds, how much of each of 77 foods should be eaten on a daily basis so that the his/her intake of nine nutrients will be at least equal to the recommended dietary allowances (RDSs) suggested by the National Research Council?

Example problem 2: the diet problem—assumptions

- Assume there are n kinds of food; the price per unit of the jth food is p_j
- Assume that there are m different nutrients, and that the nutritional requirement of nutrient i is r_i
- Finally, let a_{ij} be the amount of the ith nutrient in one unit of the jth food
- In order to rule out absurd menus we also suppose that we may limit the number of units of the jth food to u_j units

Example problem 2: the diet problem—model, I

- Variables: We define

$$x_j := \text{number of units of food } j \text{ in the diet, } j = 1, 2, \dots, n$$

- Objective function: We wish to minimize the total price of the diet, that is, the objective function, which we call f, is to

$$\text{minimize } f(\mathbf{x}) := p_1 x_1 + p_2 x_2 + \cdots + p_n x_n = \sum_{j=1}^{n} p_j x_j$$

Example problem 2: the diet problem—model, II

- Constraints There are three types of constraints:
  - Nutritional requirements Thanks to proportionality, we may write each nutritional requirement i in the form: $\sum_{j=1}^{n} a_{ij}x_j \geq r_i$. The complete set of nutritional constraints then is:

$$\sum_{j=1}^{n} a_{ij}x_j \geq r_i, \qquad i = 1, \dots, m$$

  - Food limits The diet includes an upper bound on the number of units of each food:

$$x_j \leq u_j, \qquad j = 1, \dots, n$$

  - Physical/logical

$$x_j \geq 0, \qquad j = 1, \dots, n$$

Example problems: Classification

- The staff planning problem Integer programming problem
- The diet problem Linear programming problem (continuous)
- This course focuses on continuous problems; integer programming is treated in the Applied optimization and Project courses as well as in a course given at Computer science

Modelling practice

The below figure illustrates several issues in the modelling process

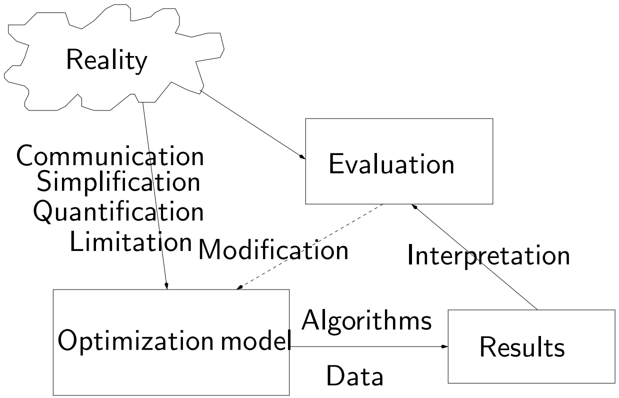


Figure: Flow chart of the modelling process

Difficulties

- Communication can often be difficult (the two parties speak different languages in terms of describing the problem)
- Problems with data collection:
  - Quantification difficult
  - Enough accuracy obtained?
  - Uncertainties (sometimes part of the problem, sometimes not)
- Conflict between problem solvability and problem realism
- Problems with the result:
  - Interpretation of the result must make sense to users
  - Must be possible to transfer the solution back into the "fluffy" world where the problem came from

Problem classification, I: General problem

$$\begin{aligned}
\mathbf{x} \in \mathbb{R}^n &: \text{vector of decision variables } x_j, \quad j = 1, 2, \ldots, n \\
f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\} &: \text{objective function} \\
X \subseteq \mathbb{R}^n &: \text{ground set defined logically/physically} \\
g_i : \mathbb{R}^n \to \mathbb{R} &: \text{constraint function defining restriction on } \mathbf{x}:
\end{aligned}$$

$$\begin{aligned}
g_i(\mathbf{x}) &\geq 0, \qquad i \in \mathcal{I} \quad \text{(inequality constraints)} \\
g_i(\mathbf{x}) &= 0, \qquad i \in \mathcal{E} \quad \text{(equality constraints)}
\end{aligned}$$

Problem classification, I: General problem

The optimization problem then is to

$$\begin{array}{lll} \underset{\mathbf{x}}{\text{minimize}} & f(\mathbf{x}), & \\ \text{subject to} & g_i(\mathbf{x}) \geq 0, & i \in \mathcal{I}, \\ & g_i(\mathbf{x}) = 0, & i \in \mathcal{E}, \\ & \mathbf{x} \in X & \end{array}$$

(If it is really a maximization problem, then change the sign of f)

Example problems, I

(LP) Linear programming Objective function linear:
$f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}}\mathbf{x} = \sum_{j=1}^{n} c_j x_j$ ($\mathbf{c} \in \mathbb{R}^n$);
constraint functions affine: $g_i(\mathbf{x}) = \mathbf{a}_i^{\mathrm{T}}\mathbf{x} - b_i$
($\mathbf{a}_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, $i \in \mathcal{I} \cup \mathcal{E}$);
$X = \{\, \mathbf{x} \in \mathbb{R}^n \mid x_j \geq 0, \quad j = 1, 2, \dots, n \,\}$

(NLP) Nonlinear programming Some function(s) f, g_i ($i \in \mathcal{I} \cup \mathcal{E}$) are nonlinear

Example problems, II

Continuous optimization f, g_i ($i \in \mathcal{I} \cup \mathcal{E}$) are continuous on an open set containing X; X is closed and convex

Integer programming $X \subseteq \{0, 1\}^n$ or $X \subseteq \mathbb{Z}^n$ (n-vector of integers)

Unconstrained optimization $\mathcal{I} \cup \mathcal{E} = \emptyset$; $X = \mathbb{R}^n$

Constrained optimization $\mathcal{I} \cup \mathcal{E} \neq \emptyset$ and/or $X \subset \mathbb{R}^n$

Example problems, III

Differentiable optimization f, g_i $(i \in \mathcal{I} \cup \mathcal{E})$ are at least once continuously differentiable on an open set containing X (that is, "in C^1 on X," which means that ∇f and ∇g_i exist there and the gradients are continuous); further, X is closed and convex

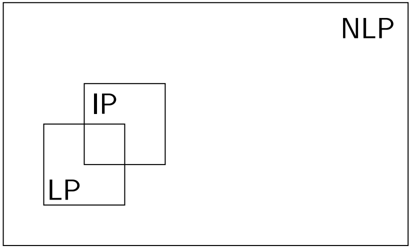
Non-differentiable optimization At least one of f, g_i $(i \in \mathcal{I} \cup \mathcal{E})$ is non-differentiable

(CP) Convex programming f is convex; g_i $(i \in \mathcal{I})$ are concave; g_i $(i \in \mathcal{E})$ are affine; X is closed and convex

Non-convex programming The complement of the above

Problem relations

Relations among NLP, IP, and LP:



- LP special case of NLP: a linear function is a special kind of nonlinear function (cf. Taylor expansion)
- IP special case of NLP: $x_j \in \{0, 1\}$ equivalent to $x_j(1 - x_j) = 0$

Some IP problems are equivalent to LP—*integrality property*. (Example: The shortest path problem)

Rough distinctions between LP and NLP

LP Linear programming $\approx$ applied linear algebra. LP is "easy," because there exist algorithms that can solve every LP problem instance efficiently in practice

NLP Nonlinear programming $\approx$ applied analysis in several variables. NLP is "hard," because there does *not* exist an algorithm that can solve every NLP problem instance efficiently in practice. NLP is such a large problem area that it contains very hard problems as well as very easy problems. The largest class of NLP problems that are solvable with some algorithm in reasonable time is CP (of which LP is a special case)

On the material in this course

- If there are no $\geq$- or $\leq$-constraints then the problem is essentially unconstrained
- $=$-constraints are treated through numerical analysis techniques. So, unconstrained optimization is essentially a numerical analysis subject
- With $\geq$- or $\leq$-constraints we face problems such as which are the active constraints. One-sidedness
- Results in difficult "non-differentiabilities"
- Largely a subject of convex and variational analysis. This is therefore a major part of this basic course

Computer communication, I

- Problem statement: connect computers so that they can all communicate; minimize the total length of the cables
- These connections are known as spanning trees—hence we wish to solve the *minimum spanning tree (MST)* problem

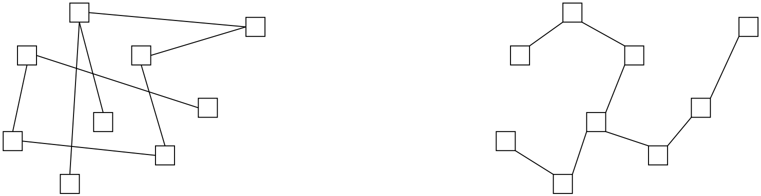


Figure: A non-optimal (left) and optimal (right) spanning tree

- If # computers $= n$, then # possible connections $= n^{n-2}$

Computer communication, II

- "Method:" enumerate them all, compare. Suppose it takes 10^{-9} s. to evaluate one spanning tree

n	total time
10	0.1 s.
15	22.5 days
20	8.3 million years

- MST can be solved in a time proportional to $n \log n$; it is hence "easy" (the complexity term is "polynomial")

The traveling salesman problem (TSP), I

- Problem statement: visit n cities in an order which minimizes the total distance traveled, then return to the initial city; do not revisit any other city
- Interesting in that it has many practical applications: vehicle routing, paper cutting, job sequencing on single machines, ...
- Total number of traveling salesman tours ("Hamilton cycles") is $n!$; similar combinatorial explosion as in MST. Does there exist an efficient algorithm?
- No! (Unless $\mathrm{P} = \mathrm{NP}$: an unsolved Millenium problem)
- Heuristics often used for large-scale examples

The traveling salesman problem, II

- Practical application: masters project at LiU 1988 with Philips, Norrköping
- Philips produce hundreds of circuit boards per day in batches
- The drilling machine is connected to a microcomputer that selects the ordering of the holes to be drilled, given their coordinates
- Algorithm: a simple sorting operation—for every fixed x-coordinate, the y-coordinates are sorted in increasing order
- Takes too long to drill one circuit board (the path is too long)

Before

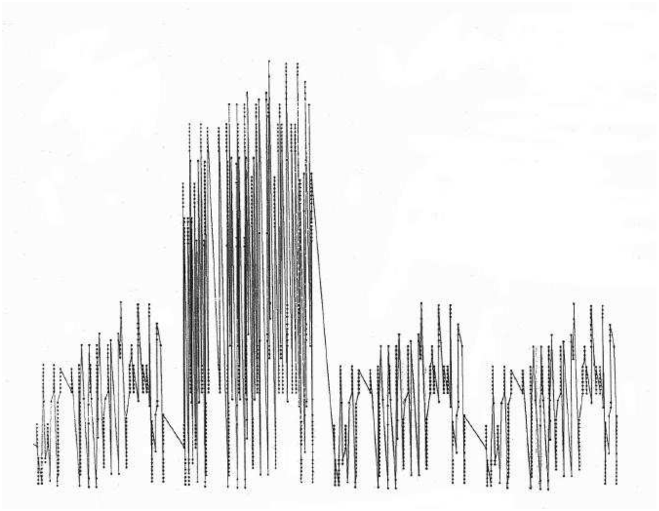


Figure: Initial drill pattern

The traveling salesman problem, III

- This is a TSP problem! Each hole corresponds to a city which is to visited = hole to be drilled exactly once
- The masters students implemented a heuristic algorithm based on Lagrangian duality and solutions of MST problems, which produces a feasible solution quickly (Section 6.7.2)
- Moreover, the algorithm produced bounds on the solution such that one gets a quality measure
- Example: the optimal path is around 2 meters long, and the heuristic solution provides a drilling pattern no more than 7 % longer than an optimal one
- Result: Philips could increase their production by about 70 %
- The theory behind the computational technique is a subject of the Project course in optimization

After

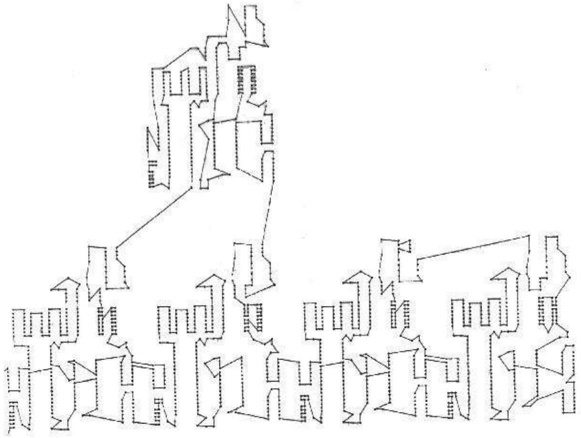


Figure: Near-optimal drill pattern